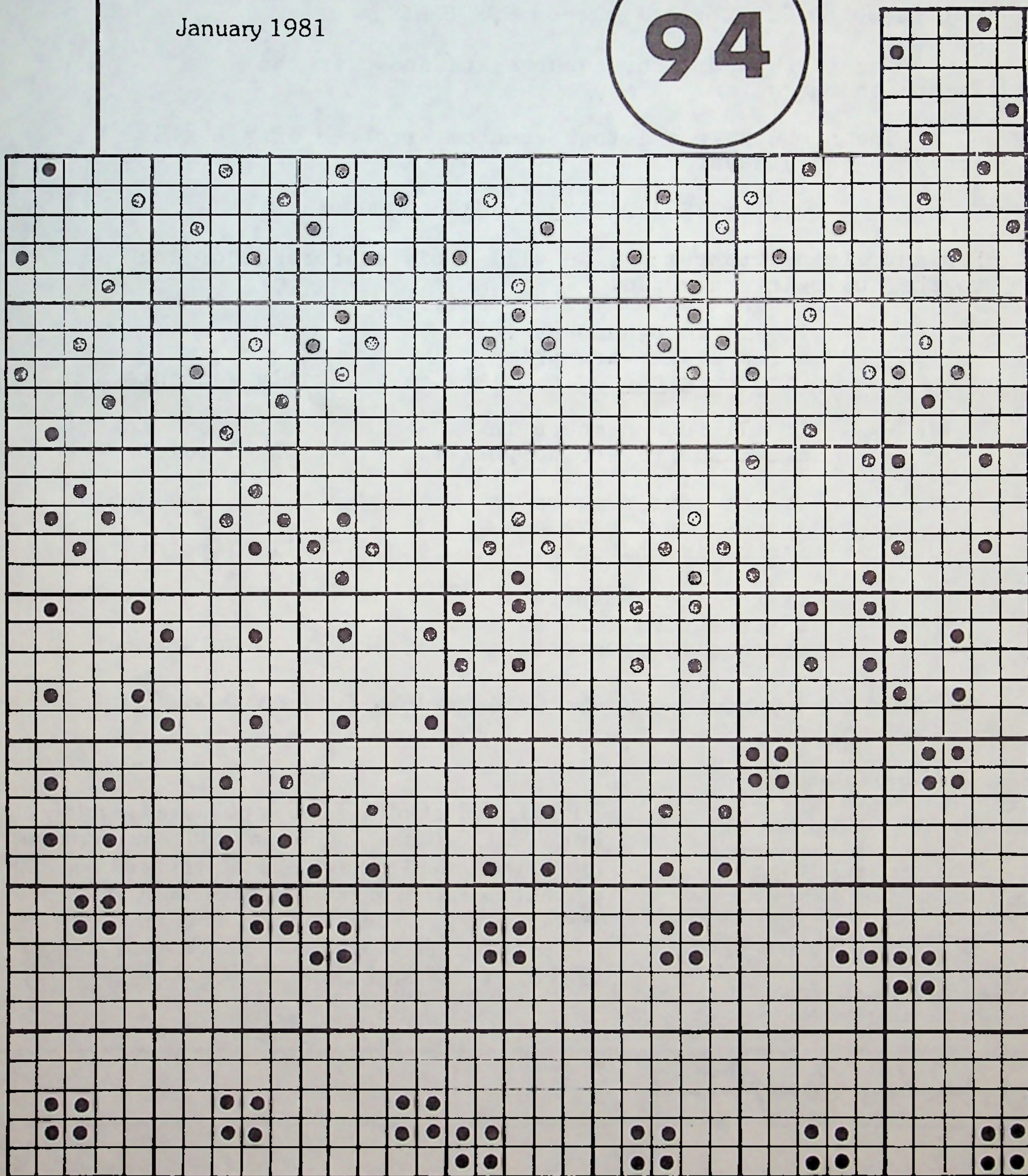
Popular Computing

Volume 9 Number 1

January 1981

94



Fivesquare

In a 5 x 5 array, there are exactly 50 sets of four squares that themselves form a square (connecting their centers). This agrees with the formula on page 6 of our issue 89 (in the article on NOSQUARE).

The 50 sets of four squares are shown in the Figure on the cover.

The numbers from 1 to 25 can be arranged in a 5 x 5 array in (25!) ways:

= 15511210043330985984000000.

For any given arrangement, we will apply a score according to the following schedule:

If all four numbers in a sub-square are prime (counting one as a prime). . . .	tally 4 points
If all four numbers in a sub-square are composite. . .	tally 1 point
If the four numbers in a sub-square have a factor in common	tally 1 point
If the four numbers in a sub-square form an arithmetic progression . . .	tally 10 points

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $20.50 per year, or $17.50 if remittance accompanies the order. For Canada and Mexico, add $1.50 per year. For all other countries, add $3.50 per year. Back issues $2.50 each.

1	1	2	6	7
2	2	3	7	8
3	3	4	8	9
4	4	5	9	10
5	6	7	11	12
6	7	8	12	13
7	8	9	13	14
8	9	10	14	15
9	11	12	16	17
10	12	13	17	18
11	13	14	18	19
12	14	15	19	20
13	16	17	21	22
14	17	18	22	23
15	18	19	23	24
16	19	20	24	25
17	2	6	8	12
18	3	7	9	13
19	4	8	10	14
20	7	11	13	17
21	8	12	14	18
22	9	13	15	19
23	12	16	18	22
24	13	17	19	23
25	14	18	20	24
26	1	3	11	13
27	2	4	12	14
28	3	5	13	15
29	6	8	16	18
30	7	9	17	19
31	8	10	18	20
32	11	13	21	23
33	12	14	22	24
34	13	15	23	25
35	1	4	16	19
36	2	5	17	20
37	6	9	21	24
38	7	10	22	25
39	1	5	21	25
40	3	6	17	14
41	4	7	18	15
42	8	11	22	19
43	9	12	23	20
44	3	11	15	23
45	3	12	19	10
46	2	11	18	9
47	8	17	24	15
48	7	16	23	14
49	2	16	24	10
50	4	6	22	20

The 50 squares. Numbering the cells in the array from 1 to 25, the table shown here lists the combinations of four cells that make up the sub-squares.

2	25	13	9	23
18	15	14	6	10
21	24	5	12	17
8	3	16	7	1
19	4	20	11	22

28

8	15	12	11	6
2	19	5	16	20
9	3	7	1	10
25	22	17	13	23
24	18	4	14	21

47

3	22	20	8	23
16	14	24	2	21
15	13	7	4	17
10	9	11	5	18
6	19	1	25	12

50

4	16	24	2	14
12	18	8	9	1
21	20	5	6	17
22	10	7	13	3
25	15	23	19	11

54

2	6	24	23	11
12	14	18	19	17
8	10	1	20	7
15	16	21	3	13
25	9	22	5	4

65

15	21	9	7	23
24	20	8	10	12
5	14	16	3	13
11	4	2	19	22
1	6	18	17	25

52

4	14	21	5	7
9	8	6	19	2
20	12	25	18	15
16	24	3	13	22
23	17	10	11	1

39

Test data for the FiveSquare problem. The score for each test array is given next to it.

PROBLEM 283

Problem: What arrangement will then produce the greatest score?

The array represented by A(I) = I (that is, the array with the numbers in normal order) has a score of 34, consisting of 3 squares of primes, 12 squares with a factor in common, and 10 squares made up of composite numbers.

Other test arrangements are given. An arrangement has been found with a score of 122.

Random arrangements of the numbers in the cells of the array can be created by any of the standard methods of creating random permutations (a dozen possible schemes were presented in our issue number 50). A quick and dirty scheme is the following:

1. Fill the array by A(K) = K.

2. Generate a random integer in the range from 1 to 25; call it R.

3. Interchange element A(I) of the array with element A(R).

4. Repeat steps 2 and 3 25 times, with I running from 1 to 25.

In terms of BASIC, the following subroutine will implement the above scheme, where RND(1) returns a fraction between zero and one.

```
2000  FØR K = 1 TØ 25
2010  A(K) = K
2020  NEXT K
2030  FØR K = 1 TØ 25
2040  R = INT(25*RND(1) + 1)
2050  TEMP = A(K)
2060  A(K) = A(R)
2070  A(R) = TEMP
2080  NEXT K
2090  RETURN
```

Stages	First Occurrence	Distribution
1	100000	901
2	100001	8031
3	100011	30179
4	100026	68826
5	100039	113017
6	100077	148529
7	100117	158301
8	100139	139609
9	100429	103073
10	100529	64549
11	100777	35819
12	101587	17542
13	104815	7451
14	103698	2879
15	123998	986
16	238521	271
17	331809	102
18	583895	24
19	691845	2

1	10000000	9001
2	10000001	117092
3	10000011	580092
4	10000026	1721776
5	10000039	3777752
6	10000077	6659395
7	10000117	9774369
8	10000139	12210996
9	10000429	13165051
10	10000529	12418609
11	10000777	10373834
12	10001117	7704328
13	10003669	5143332
14	10004929	3107025
15	10008539	1705741
16	10014765	859351
17	10014283	396085
18	10027953	169610
19	10017790	67874
20	10014944	25290
21	10128033	8916
22	12018433	3092
23	15418423	996
24	31789816	318
25	41279617	62
26	70319091	8
27	83099616	4
28	88169426	2

Problem 280 (The Six-Digit Algorithm) in issue number 92 called for partitioning a 6-digit number into two 3-digit parts; multiplying those parts together to produce a new 6-digit number; and continuing this process until it degenerates to zero, and counting the number of stages.

Harry L. Nelson, Livermore, California, calculated all the results for both 6- and 8-digit numbers, as shown here.

To: Popular Computing:

Popular Computing number 92, page 3, refers to a Professor Donald Knuth who wrote a book called The Art of Computer Programming, Vol. 3, that "omits bubble completely."

I wonder which Professor Donald Knuth and which book you are referring to, since my copy of a book with the same name has the following entry in its index:

> Bubble sort, 106-111, 134-135, 141, 223-224, 244-246, 352-353, 360, 379, 386, 388, 644.

Of course, something that is mentioned on only 22 pages is easy to miss.

The method described in that book also bears a curious resemblance to the flowchart given on page 7 of PC92, except that the book method is slightly more efficient and the program is shorter.

Well, I never did think many people could understand the inscrutable language of The Art of Computer Programming, but I wish there were a way to spread the word that it does have an index. This index can be used (as Jonathan Swift said) by people who wish to pretend they have read the book.

Cordially,

Donald E. Knuth

Actually, our blunder in misquoting the master is, indirectly, a compliment. We don't have a copy of Knuth, Vol. 3. Editors can obtain almost any book in their field for the asking; publishers of books could never risk refusing a request for a review copy. There is one big exception; namely, Addison-Wesley and the three volumes of Knuth--if you want them, you pay full price, no matter who you are. We will probably buy Volume 3 when Volume 4 comes out.

The blunder was unintentional, but its effect--a nice letter from Prof. Knuth--was delightful.

homework 9

On the facing page there is empirical data of 16 observed values (points 7 through 22). You are to predict, as well as you can, the time for point 29.

(1) You will, of course, want to plot the data. Will the nature of the problem be altered if you change the scale of the dependent variable?

(2) Will you try a least squares curve fit? If so, what degree seems appropriate?

(3) Could you use the method of undetermined coefficients? If so, what degree of polynomial would be appropriate?

(4) Could the problem be solved graphically, simply by making a large and accurate plot of the data?

(5) Just how accurate is the given data? What confidence do you have in the extrapolated result? In other words, how many digits of your result would you be willing to guarantee?

(6) If you use the method of least squares, then you should agree that there should be just one correct result and everyone in your class should achieve the same result (to the same precision level). Is this statement acceptable?

7	10:27:20
8	10:27:48
9	10:28:37
10	10:29:50
11	10:31:28
12	10:33:53
13	10:37:00
14	10:41:15
15	10:46:30
16	10:53:18
17	11:01:15
18	11:11:15
19	11:23:00
20	11:37:00
21	11:53:05
22	12:12:40
..	. . . .
..	. . . .
..	. . . .
29	????????

The times shown in the second column are of successive observations, in hours, minutes, and seconds, of a computing experiment conducted one morning. We seek an estimate of the time for the 29th observation.

PROBLEM 284

CURVE FITTING and EXTRAPOLATION

Progressive Mantissas

Consider the following algorithm. Start with any number, say 2, and call it A. Take its square root and add the mantissa of the square root to A.

Repeat this process indefinitely. Keep a count, N, of the number of terms and calculate the ratio N/A = Q. Thus, we develop a sequence that begins:

N	Progressive sum	Q
1	2.41421356	.414
2	2.96798754	.674
3	3.69077236	.813
4	4.61191066	.867
5	4.75944661	1.0505

Now, an average mantissa should be around .5, so the values of A should increase half as fast as N, and the ratio Q should tend toward 2.00. But it doesn't--why?

For various starting values, we have the following progressive sums and ratios at 1000 terms:

Starting A	Progressive sum	Q
2	259.345590	3.8559
3	238.561512	4.1918
5	228.559193	4.3752
7	258.210541	3.8728
10	256.054038	3.9054
12	256.748623	3.8949
15	257.346720	3.8858
20	241.371066	4.1430
22	274.219195	3.6467
24	259.645732	3.8514
26	257.556785	3.8826
28	227.984940	4.3863
30	258.925925	3.8621

PROBLEM 285

Problem Solution

Problem 195, REVERSE, in issue 55, was this:

Start with all the natural numbers, and reverse every group of K = 2, producing the sequence:

2, 1, 4, 3, 6, 5, 8, 7, 10, 9, 12, 11,...

Print the first set of 2 and delete them from the sequence. Increase K to 3. Reverse every group of 3, to produce:

6, 3, 4, 7, 8, 5, 12, 9, 10, 13, 14, 11,...

Print the first set of 3 and delete them. Increase K to 4, and reverse each group of 4, to produce the sequence:

12, 5, 8, 7, 14, 13, 10, 9, 16,...

Increase k, and so on. Calculate the first 1000 numbers to be extracted by this process.

The problem lends itself nicely to calculation in BASIC. The logical steps are these:

1. Create an array, B, of dimension 2000. (and an auxiliary array, C, also of 2000 terms)

2. Fill array B with consecutive numbers; that is, set B(I) = I.

3. Set K = 2.

4. Reverse all groups of K terms.

5. Print the first group of K terms; namely, B(1), B(2), B(3),...,B(K).

6. Shift the array left K terms.

7. Increment K by 1.

8. Go back to step 4 and repeat.

In terms of Applesoft BASIC, some of these steps can be carried out as follows:

Step 4.

```
1000 FØR I = 1 TØ 2000
1010 C(I) = B(I)
1020 NEXT I

1030 FØR J = 1 TØ 1900 STEP K
1040 FØR L = 1 TØ K
1050 B(J+K-L) = C(J+L-1)
1060 NEXT L
1070 NEXT J

1080 RETURN
```

Step 5.

```
2000 FØR I = 1 TØ K
2010 PRINT I, B(I)
2020 NEXT I

2030 RETURN
```

Step 6.

```
3000 FØR I = 1 TØ 1900
3010 B(I) = B(I+K)
3020 NEXT I

3030 RETURN
```

The first four successive groups that are printed are:

1	2	6	12	16
2	1	3	5	9
3		4	8	10
4			7	13
5				14

and the next seven groups are these:

1	22	32	48	52	58	84	94
2	15	25	31	35	51	67	77
3	18	26	38	42	54	68	90
4	11	19	21	29	41	61	57
5	24	20	28	36	44	62	74
6	17	23	27	39	37	45	47
7		30	34	40	60	46	64
8			33	49	43	55	63
9				50	66	56	70
10					53	59	69
11						76	86
12							85

Examination of these, and successive groups, shows a pattern. If we list, for each group, the number of the largest term and the number of the smallest term, we get the accompanying table.

Will this pattern continue indefinitely? That is, can we say with assurance that when we reverse groups of 1002 terms, the 1002 numbers that are extracted and printed will have their largest number at the 1001st term and the smallest number at the 502nd term?

Incidentally, the group for K = 45 will print out as follows:

1352	1205	1218	1149	1150	1143	1144	1107
1108	1055	1062	1045	1046	931	932	895
896	889	890	821	834	795	828	831
832	857	874	909	942	961	968	971
978	981	1070	1111	1112	1115	1138	1157
1210	1229	1236	1245	1282	1353		

Thus, the 1000th number to be printed is 1062.

PROBLEM 286

2	1	2	26	25	14
3	1	2	27	1	14
4	1	2	28	1	14
5	1	2	29	1	14
6	5	4	30	29	16
7	1	4	31	1	16
8	1	4	32	1	16
9	1	4	33	1	16
10	9	6	34	33	18
11	1	6	35	1	18
12	1	6	36	1	18
13	1	6	37	1	18
14	13	8	38	37	20
15	1	8	39	1	20
16	1	8	40	1	20
17	1	8	41	1	20
18	17	10	42	41	22
19	1	10	43	1	22
20	1	10	44	1	22
21	1	10	45	1	22
22	21	12	46	45	24
23	1	12	47	1	24
24	1	12	48	1	24
25	1	12	49	1	24
			50	49	26

This is K

In group K, this is the number of the term with the highest value

In group K, this is the number of the term with the lowest value.

A pattern observed in the REVERSE problem.

Computer Economics

The following square roots of integers:

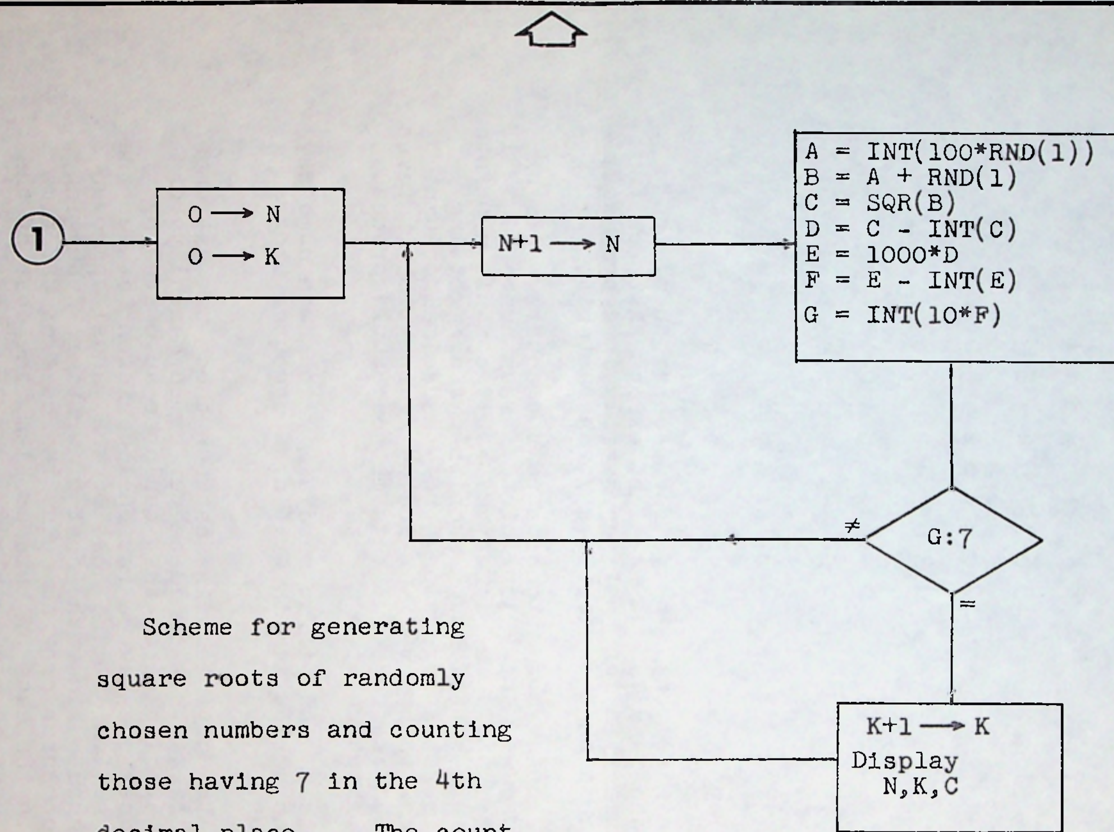
7	2.64575131
31	5.56776436
37	6.08276253
42	6.48074069
58	7.61577310
76	8.71779788
78	8.83176086
95	9.74679434

have a feature in common; namely, for each of them, the 4th decimal place is a 7. We might expect that, on the average, one number in ten (not necessarily integers) would have a square root that exhibited that characteristic.

But restricting our attention to the square roots of integers: suppose we want a long list of such numbers. If the long list is only a few dozen, we would consult a printed table of square roots (although such things are getting scarce); put a straightedge along the 4th decimal column; and read off results by eye. Including the time it takes to record the numbers we seek, this method should proceed at the rate of about one per minute.

On the other hand, if we need a million such numbers, we would have no hesitation in writing a computer program to sift them out.

Where is the dividing line between those two extremes? There are many tasks that could be performed manually (or semi-manually) by clerks, or that could be automated by computer; how does one make a rational decision between the two courses of action? The cost of the manual approach is usually weighed against the cost of writing, debugging, and testing a computer program, plus the cost of CPU time (although, as in the case of this problem, the cost of the computer itself may be negligible).



Scheme for generating square roots of randomly chosen numbers and counting those having 7 in the 4th decimal place. The count, K, should be approximately 1/10 of the total count, N.

Suppose we make the following assumptions:

(1) Programmers cost four times as much as clerks.

(2) The problem can be analyzed, coded, debugged, tested, and production begun, in four hours.

If we go the computer route, then after the program is in production, the cost per unit result will approach zero the longer we run the program. If we go the clerk route, the total cost will be less up to 16 clerk-hours, so whatever amount can be done in that time is the cutoff value. The trivial task we have outlined would best be done by two clerks working together; in 16 clerk-hours we might expect to get 500 correct results.

PROBLEM 287

The problem involved here suggests two further problems:

(1) Considering the square roots of successive integers, what is the longest string of consecutive <u>integers</u> that one can find for which each square root does NOT have 7 in the 4th decimal place?

(2) For <u>any</u> numbers, it is to be expected that about one in ten square roots will have a 7 in the 4th decimal place. The accompanying flowchart suggests a way to demonstrate this. The functions on the flowchart are those of BASIC; namely, INT for the greatest integer in a number, and RND(1) for a new random number between zero and one.

Cycle Lengths

It is well known that the low-order digits of powers of 2 cycle as follows:

1 0 4 8 5 7 6

by 4's

by 20's

by 100's

by 500's

by 2500's

A similar pattern is exhibited by powers of 3:

the unit's digit cycles by	4's
the low-order two digits cycle by	20's
the low-order three digits cycle by	100's
the low-order four digits cycle by	500's
the low-order five digits cycle by	2500's

...and so on...

Powers of 5 have a distinctly different pattern. After the first power, <u>all</u> powers end in 25, and the cycle length for the R low-order digits is as follows:

R	L
1	1
2	1
3	2
4	4
5	8
6	16

The table shown here gives the first 25 powers of 2 and 3 (that is, their low-order three digits) and sufficient powers of 5 to demonstrate the repetition in the low-order digits.

	2	3	5
1	2	3	5
2	4	9	25
3	8	27	125
4	16	81	625
5	32	243	3125
6	64	729	15625
7	128	187	78125
8	256	561	90625
9	512	683	53125
10	024	049	65625
11	048	147	28125
12	096	441	40625
13	192	323	03125
14	384	969	15625
15	768	907	
16	536	721	
17	072	163	
18	144	489	
19	288	467	
20	576	401	
21	152	203	
22	304	609	
23	608	827	
24	216	481	
25	432	443	

...one might conjecture that the cycle length for the seven low-order digits of powers of 5 is 32. But conjectures like that are dangerous.

And what of powers of other numbers? Powers of 7 seem to have the same pattern in their cycle lengths as powers of 2, but powers of 11 are quite different:

R	L
1	1
2	10
3	50

It would seem that there is room here for a modest piece of research, which could be done on any machine in any language (the results given above were obtained in BASIC).

PROBLEM 288

In the book Programming in BASIC for Personal Computers (David Heiserman, Prentice-Hall) there appears the following amazing paragraph:

> The function EXP(n) simply raises any number n to the *e*th power, where *e* is a constant value very close to 2.71828. The same sort of function can be carried out by means of the BASIC formula n↑2.71828. But, of course, EXP(n) is far easier to use and it runs much faster on the computer. Again, this is one of those functions normally reserved for engineering and scientific applications.

Is there any value of n for which this weird distortion could be true? That is, for what value of n, if any, is it true that:

$$e^n = n^e \ ?$$

Doubtless this question can be settled analytically, but it is posed here as a computing problem.

On the assumption that there is a value of n to be found, set up a search procedure (bracketing would be ideal) for it. If your search fails, can you then condlude that there is no value of n that satisfies?

PROBLEM 289